하드웨어 복호화기 구현을 고려한 VVC 부호화 도구 및 제약 분석

ㅁ 이상헌 / LG전자

요약

고해상도 영상 서비스 증가와 Youtube, Netflix와 같은 OTT 중심의 동영상 시청 환경 변화에 따라, 기존 대비 높은 압축률을 목표로 하는 새로운 비디오 코덱 표준들이 활발히 개발되고 있다. 그 중 가장 대표적인 것은 MPEG에서 개발 중인 VVC(Versatile Video Coding)로 기존 HEVC(High Efficiency Video Coding) 대비 2배의 압축률 달성을 목표로 하고 있다. 하지만 압축률 향상을 위한 새로운 부호화 도구들의 복잡도가 점점 증가함에 따라, 하드웨어 복호화기 구현시 많은 어려움이 있을 것으로 예상된다. VVC 표준화에서는 이런 어려움을 극복하기 위하여, 하드웨어 구현 관점에서 부호화 도구들을 최적화하고 이를 사용함에 있어 몇 가지 제약들을 정의하고 있다. 본 논문에서는 현재 VVC CD[1] 및 VTM 6.0 소프트웨어[2] 기준으로 하드웨어 복호화기 구현을 고려하여 채택된 부호화 도구 및 제약에 대하여 분석한다.

Ⅰ. Introduction

반도체 미세 공정의 발전으로 인하여, 압축률 향상을 위한 복잡한 부호화 도구들을 하드웨어 복호화기로 구현할 때 발생할 수 있는 많은 기술적 한계들은 해결되고 있다. 하지만, 하드웨어 복호화기들이 대량 생산을 하는 TV와 핸드폰 같은 기기들에 SoC 형태로 탑재된다는 점을 고려하면, 하드웨어 복호화기를 작은 면적으로 설계하고 발열 및 배터리 소모량을 최소화할 수 있도록 적절한 동작 클럭(Clock)을 선택하여 설계하는 것은 매우 중요하다. 또한, 영상 해상도 증가에 따라 SoC에서 사용하는 메모리 대역폭(Memory BandWidth)이 급증한다는 점을 고려하면 복호화기가 사용하는 메모리 대역폭을 줄이는 것도 매우 중요하다. 이를 위해, VVC 표준에서는 하드웨어 복호화기의 면적과 동작

Entropy Decoder	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6
Prediction		Block 0	Block 1	Block 2	Block 3	Block 4	Block 5
Transform		Block 0	Block 1	Block 2	Block 3	Block 4	Block 5
Reconstruction			Block 0	Block 1	Block 2	Block 3	Block 4
Loop Filter				Block 0	Block 1	Block 2	Block 3

〈그림 1〉 하드웨어 비디오 복호화기 파이프라인 구성 예

클럭 그리고 메모리 대역폭을 최소화할 수 있도록 부호화 도구를 최적화하고, 압축 성능에서 손실이 있더라도 하드웨어 개발에 도움이 되는 제약들을 표준으로 정의하고 있다.

본 논문에서는 위에서 서술된 하드웨어 면적, 동작 클럭, 그리고 메모리 대역폭을 최적화하기 위하여 현재 VVC 표준에서 채택된 하드웨어 구현 관련 부호화 도구 및 제약에 대하여 상세히 살펴보고 하드웨어 구현 측면에서 어떤 의미가 있는지를 설명한다. 그리고, 현재 VVC 표준의 하드웨어 구현 가능성 및 향후 시장 적용 전망에 대한 개인적인 의견을 제시하면서, 본 논문을 마무리한다.

II. 하드웨어 면적을 고려한 특성

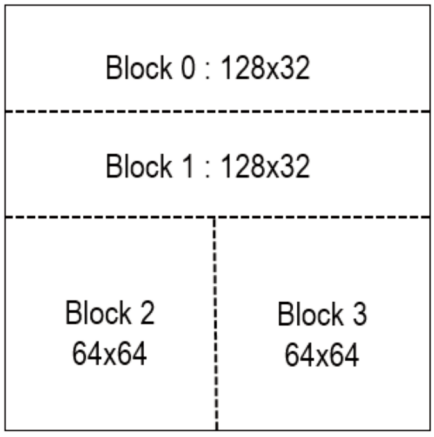
VVC는 HEVC 대비 2배의 압축 성능 향상을 위하여 새로운 부호화 도구들을 대거 채택하였고, CTU 크기도 기존 HEVC 대비 4배 증가한 128x128을 사용한다. 따라서, 기존 표준 대비 하드웨어 복호화기의 면적 증가는 불가피할 것으로 예상된다. 따라서, VVC 표준 진행 과정에서 압축률을 크게 손해 보지 않으면서 하드웨어 구현 면적을 의미 있게 감소시킬 수 있는 다양한 방법들이 활발히 논의되었고, 그 결과들이 표준에 적용되어 있다. 본 장에서는 이와 관련된 대표적인 특징 및 개념에 대해서 상세히 설명한다.

1. VPDU 개념 도입

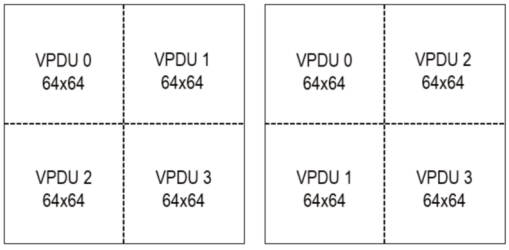
일반적으로, 하드웨어 비디오 복호화기를 구현할 때는 주요 부호화 도구들을 기능 및 상관 관계를 고려하여 분할한 후, 〈그림 1〉과 같은 파이프라인(pipeline)으로 구성하여 병렬 처리하도록 설계한다. 이 때, 각 파이프라인 사이의 버퍼 크기가 하드웨어 면적에 큰 비중을 차지하는데 이는 파이프라인 최대 동작 블록 크기에 의해 결정된다. 하드웨어 복호화기 구현에서 파이프라인의 최대 블록 크기는 Transform 블록에 의해 결정되는데, 이는 다른 블록들의 경우 주어진 블록을 임의의 작은 블록으로 분할하여 동작하도록 설계하는 것이 가능하나, Transform 블록의 경우 이런 방법의 적용이 불가능하기 때문이다.

VVC에서는 최대 64x64 블록의 Transform을 사용하므로, 하드웨어 복호화기에서는 64x64 블록 크기의 파이프라인을 사용해야 한다. VVC 표준에서는 이를 VPDU(Virtual Pipeline Data Unit)라는 개념으로 정의하고, 분할된 블록은 하나의 VPDU 이내에 완전히 존재하거나, 다른 VPDU에 걸쳐서 존재할 경우, 해당 블록이 그 VPDU를 완전히 차지해야 된다는 제약을 정의하고 있다. 따라서, VPDU를 고려하지 않는다면, 〈그림 2〉와 같은 블록 분할

이 가능할 수 있지만 현재 표준에서는 〈그림 2〉와 같은 블록 분할은 불가능하고, 〈그림 3〉과 같은 형태의 VPDU 및 블록 순서만 발생 가능하다.



〈그림 2〉 VVC에서 불가능한 블록 분할



〈그림 3〉 VVC 에서 가능한 VPDU 모양 및 순서

이는 하드웨어 복호화기 구현상 편의를 제공하기 위한 제약으로, 다양한 파이프라인 입력 블록 모양 및 순서를 제어하기 위해 발생할 수 있는 버퍼 크기 및 구현 복잡도 증가를 감소시킨다. 또한, 이런 제약이 없다면 하드웨어 설계 방법에 따라서는 128x128 파이프라인 버퍼 선택이 불가피할 수 있으며, 이는 하드웨어 면적 증가를 가져올 수 있다.

2. 라인 버퍼(Line Buffer) 최적화

1) CTU Row Boundary 제약

비디오 코덱 표준들은 윗 블록과 현재 블록간의 데이터 상관 관계를 활용하기 때문에, 하드웨어 복호화기는 라인 버퍼를 가지고 있어야 한다. 일반적으로 라인 버퍼는 하드웨어 설계에서 SRAM으로 구현되며, 그 크기는 지원 해상도의 최대 너비에 비례한다. 최근 영상 기기들이 최대 8K 해상도까지 지원한다는 점을 고려하면 라인 버퍼가 복호화기 면적에서 큰 비중을 차지하게 된다.

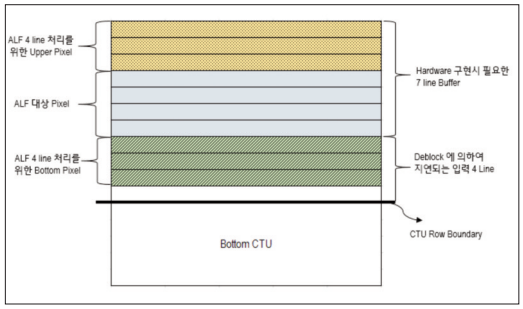
VVC에서는 압축 성능 극대화를 위해 HEVC 대비 윗 블록과의 상관 관계 활용시 더 많은 데이터들을 사용한다. 대표적인 예로 MRL(Multi-Reference Line Intra Prediction)과 Long Tap 디블럭킹 필터(Long-Tap Deblocking Filter)가 있으며, 해당 기능들은 각각 3 라인, 8 라인의 윗 블록 픽셀 정보를 활용하기 때문에 각각 1 라인, 4 라인씩만 사용하던 HEVC 대비 더 큰 크기의 라인 버퍼를 필요로 할 수 있다. 또한, VVC에서는 Affine Motion이 채택됨으로써, Affine Motion Prediction을 위한 CPMV(Control Point MV) 정보가 블록별 MV와는 별도로 추가로 필요하다. 이와 같은 특성을 고려하면 현재 VVC는 HEVC 대비 2배 이상의 라인 버퍼가 필요할 수 있으며, 이는 하드웨어 복호화기 면적에 큰 영향을 미치게 된다.

따라서, 현재 VVC 표준에서는 해당 기능들로 인하여 HEVC보다 라인 버퍼가 증가하는 것을 방지하기 위하여 CTU Row Boundary 관련 제약들이 정의되어 있다. 즉, MRL과 Long Tap 디블럭킹 필터의 경우에는 CTU Row 경계에서는 동작하지 않도록 정의되어 있다. CPMV의 경우에는, CTU Row 경계에서는 윗 블록의 CPMV를 사용하는 대

신 블록별 MV만으로 CPMV를 생성하는 방법을 사용한다. 이런 제약은 CTU 내부의 블록 경계 및 CTU Column 경계에서는 적용되지 않는다.

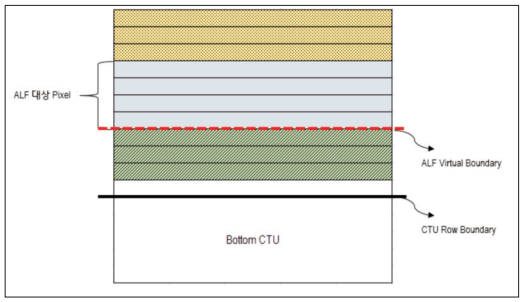
2) ALF(Adaptive Loop Filtering) virtual boundary

ALF는 VVC 표준에 채택된 신규 부호화 도구 중 가장 높은 압축 성능을 보이는 단일 부호화 도구이다. ALF는 각 4x4 블록을 기준으로 ALF 결과를 추출하기 위하여 상하좌우 각각 3 픽셀의 추가 데이터 입력이 필요하다.



〈그림 4〉 ALF 처리를 위한 라인 버퍼

따라서, 하드웨어로 구현시 〈그림 4〉와 같이 현재 파이프라인에서는 가장 아래 4x4 블록을 처리할 수 없다. 해당 4x4 블록은 그 다음 CTU Row 파이프라인을 진행할 때 처리할 수 있는데, 이 때 4x4 블록의 위쪽 3 픽셀 데이터가 필요하므로 총 7개의 라인 버퍼가 필요하다. 이는 기존 구현에서 라인 버퍼를 가장 많이 쓰는 디블럭킹 필터(4 라인 버퍼)에 비해서도 매우 많은 양이기 때문에 하드웨어 면적에 치명적인 영향을 미친다. 따라서, VVC에서는 해당 라인 버퍼를 제거하기 위하여 CTU Row 경계에서 virtual boundary라는 개념을 도입하였다.



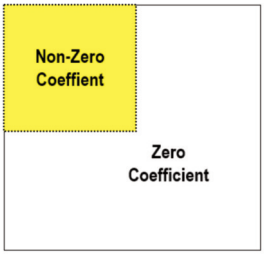
〈그림 5〉 ALF 라인 버퍼 제거를 위한 Virtual Boundary

〈그림 5〉처럼, CTU Row 경계의 4 라인 위쪽을 virtual boundary로 설정하고, 해당 경계에서는 ALF 필터를 적용할 때 4x4 블록 픽셀 데이터를 미러링(Mirroring)하여 ALF 연산을 수행한다. 따라서, 현재 파이프라인에 입력된 모든 픽셀 데이터에 대하여 출력을 만들 수 있고, 다음 CTU Row를 처리할 때도 이전 Row 파이프라인 데이터를 사용하지 않으므로 라인 버퍼를 제거할 수 있다. 물론, 〈그림 5〉에서 보듯이 ALF 파이프라인 입력이 2개 이상의 CTU에 걸쳐서 발생할 수 있으므로, CTU 단위로 전송되는 ALF 필터 세트(filter set)를 위한 라인 버퍼는 필요할 수 있다. 하지만 실제 구현시 APS(Adaptation Parameter Set)로 전송되는 모든 ALF 필터 세트가 별도의 SRAM으로 저장되어 있을 것을 고려하면, 필터 계수 값 자체를 저장하지 않고 필터 계수가 저장된 위치의 인덱스(index)만을 저장하면 되므로, 해당 라인 버퍼는 복호화기 면적에 큰 영향을 미치지 않는다.

3. Transform Zeroing Out과 LFNST

VVC에서는 MTS로 불리는 DCT-2, DST-7, 그리고 DCT-8과 같은 다양한 Primary Transform들을 지원하며, Primary Transform의 저주파수

대역에 대하여 non-separable한 Transform을 추가적으로 수행하는 LFNST를 지원한다. 이 때, 모든 Transform에 대하여 〈그림 6〉과 같이, 고주파수 대역의 계수들을 모두 다 0으로 강제하는 Zeroing-Out이라는 개념이 사용되고 있다.



〈그림 6〉 Zeroing-Out 개념

구체적으로 살펴보면, 최대 64x64의 Kernel을 사용하는 DCT-2에 대해서는 블록의 크기가 32x32보다 클 때, 저주파수 대역 32x32 영역에서만 유효한 계수가 존재할 수 있고, 최대 32x32의 Kernel을 지원하는 DST-7과 DCT-8에 대해서는 블록의 크기가 16x16보다 클 때, 저주파수 대역 16x16에서만 유효한 계수가 존재한다. 또한, LFNST 적용시에는 LFNST의 출력에 해당하는 영역에서만, 유효 계수가 존재하고 나머지 MTS 영역은 모두 0이 되어야 하는 제약이 존재한다.

이런 제약들은 하드웨어 복호화기 설계시 필요한 곱셈기(Multiplier) 수를 감소시켜주는 의미를 가지고 있다. 〈표 1〉은 DCT-2에서 32x32 이상 블록에 대하여 필요한 곱셈기의 수를 Zeroing-Out 적용 전, 적용 후로 나누어서 보여준다(편의상 버터플라이 구현은 고려하지 않음). 〈표 1〉에서 보듯이 64x64, 64x32, 그리고 32x64에 Zeroing-Out을 적용함으로써, 계수당 필요한 곱셈기의 수가 32x32와 동일하게 유지됨을 알 수 있다.

〈표 1〉 DCT-2 에서 필요한 곱셈기 수

Block 크기	곱셈기 수/계수 (Zeroing Out 미적용)	곱셈기 수/계수 (Zeroing Out 적용)
32x32	64	N/A
64x32	64	48
32x64	128	64
64x64	128	48

〈표 2〉 LFNST 적용시 LFNST와 MTS의 유효 Coefficient 수

Block 크기	유효 LFNST 입력 수	유효 MTS 입력 수
4x4	8	16
8x8	8	48
16x16	16	48
32x32	16	48

또한, DCT-2는 버터플라이 방식으로 구현하여 곱셈기 수를 줄일 수 있으나, DST-7, DCT-8에 대해서는 버터플라이를 사용할 수 없으므로 DCT-2에 비해서 최대 허용 블록 크기를 작게 제한하고(32x32), 이 때도 Zeroing-Out 을 적용함으로써, Transform 블록에서 필요한 곱셈기 수를 DCT-2를 버터플라이로 구현할 때 필요한 곱셈기 수 이하로 유지할 수 있도록 한다. 따라서, Primary Transform에 대해서는 VVC 설계시 필요한 Transform 관련 곱셈기 수가 HEVC와 동일 수준으로 유지된다.

VVC에 새로 채택된 LFNST의 경우, 복호화기 동작시 LFNST 출력을 다시 MTS로 변환 해야 되므로 추가적인 곱셈기가 필요하다. 하지만, 현재 VVC 표준에서는 LFNST 적용시 MTS 영역에 대해서 모두 다 0계수를 가지도록 강제함으로써, LFNST 수행 후 MTS를 수행할 때 유효 입력의 수가 〈표 2〉와 같이 4x4 블록을 제외하고는 기존

MTS 대비 감소하게 되어 있다. 이는, 실제 하드웨어 구현시 LFNST와 MTS를 다른 파이프라인에서 구현하지 않고 동일 파이프라인에서 기존 MTS 동작에 필요한 곱셈기를 공유해서 사용할 수 있게 하여, LFNST로 인한 추가 곱셈기 수를 최적화하는데 도움을 준다.

또한, DCT-2 적용시 64x64 블록에서 Zeroing Out을 지원하는 것은 곱셈기 수를 줄이는 것뿐만 아니라, 파이프라인 버퍼의 크기를 줄이는 효과도 기대할 수 있다. VVC 표준에서 양자화 된 계수의 범위는 -2^{15}~$2^{15}-1$의 16 비트(bit) 값으로 표현된다. 따라서, Transform 파이프라인 버퍼에는 픽셀 데이터가 저장되는 다른 파이프라인보다 더 큰 파이프라인 버퍼가 필요하다. 하지만 Zeroing-out을 통하여 유효 계수의 수가 감소하기 때문에, Transform 파이프라인의 버퍼 크기를 크게 감소시킬 수 있다.

4. DMVR과 BDOF 제약

VVC에서는 인터 블록의 움직임 보상(Motion Compensation) 성능을 강화하기 위하여 DMVR (Decoder Side Motion Vector Refinement) 및 BDOF(Bi-Directional Optical Flow)가 새로운 부호화 도구로 채택되어 있다. 여기서 우리가 주목해야 될 사실은, 해당 부호화 도구들은 큰 블록에 대해서 적용이 되더라도 이를 16x16 단위의 작은 블록으로 분할하여 동작이 가능하도록 설계되어 있다는 것이다.

예를 들어, DMVR의 경우 16x16보다 큰 블록에 대하여 선택되더라도, 16x16 단위로 SAD를 계산하여 각 16x16마다 서로 다른 MV를 가질 수 있게 되어 있다. BDOF의 경우에는 BDOF 연산을 위한 Gradient 계산시, 16x16 단위의 Boundary 입력으로 interpolation 필터가 적용된 픽셀을 사용하지 않고, 정수(integer) 픽셀을 사용하게 되어 있다. 이는 BDOF 적용시 16x16 단위로 움직임 보상을 수행하여도, BDOF 결과를 얻는 것을 가능하게 한다. 또한, 두 부호화 도구 모두 Early Termination을 위한 SAD 값 계산을 16x16 단위로 수행한다. 따라서, DMVR이나 BDOF가 큰 블록에 대해 적용되더라도, 움직임 보상을 16x16의 작은 블록 단위로 나눈 후 이를 하위 파이프라인(Sub-Pipeline)으로 구성하여 동작시키는 것이 가능하다. 일반적으로 움직임 보상을 하드웨어로 설계할 때 큰 블록을 작은 블록으로 분할해서 처리하는 것을 선호하는 가장 큰 이유는 움직임 보상을 위한 참조 (reference) 픽셀의 입력 수가 실제 블록 크기보다 훨씬 많기 때문이다. 그러므로, 큰 블록을 작은 블록으로 분할해서 동작할 수 있게 만들면 움직임 보상을 위한 입력 버퍼의 크기를 작게 만들 수 있다. 따라서, VVC 표준에서 DMVR과 BDOF를 16x16 입력만으로도 동작하게 할 수 있도록 알고리즘이 최적화되어 있는 점은 하드웨어 복호화기 구현 방법에 따라서 면적을 크게 감소시킬 수 있는 가능성을 제공해 준다.

Ⅲ. 하드웨어 동작 클럭을 고려한 특성

하드웨어 복호화기를 구현할 때는 제품에서 요구하는 최대 해상도 지원을 위한 동작 클럭을 가장 먼저 선택해야 한다. 물론, 반도체 공정의 발전으로 인하여 예전 대비 동작 클럭을 높이는 것이 용이해졌지만, 높은 동작 클럭의 사용은 하드웨어의 파워 소비를 늘리고, 이는 제품 구현시 발열 문제나 배터

리 소모량 문제를 유발하므로 적절한 동작 클럭을 선택하는 것은 매우 중요하다. 따라서, VVC 표준에서는 하드웨어 구현시 필요한 동작 클럭이 과도하게 증가하지 않도록 하는 다양한 기술들이 적용되었다.

1. 잔차 신호 부호화(Residual Coding)

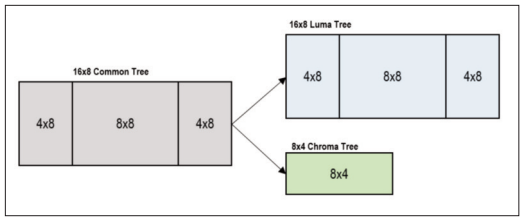
복호화기에서 동작 클럭 선택에 가장 큰 영향을 미치는 요인 중 하나는 인트라 프레임에서의 엔트로피 복호화 성능이다. 병렬 처리를 통하여 낮은 동작 클럭으로도 동작이 가능한 다른 블록과 달리, 엔트로피 복호화 블록은 CABAC 특성상 병렬화가 불가능하여 프레임당 발생하는 Bin량에 의해서 그 성능이 결정된다. 일반적으로 인트라 프레임에서 상대적으로 많은 Bin이 발생하고, 이런 경향은 표준이 발전함에 따라서 더욱 심화되므로, 인트라 프레임에서의 엔트로피 복호화 성능이 동작 클럭 결정에 가장 큰 영향을 미친다. 실제 구현에서는 이런 문제를 극복하기 위하여, 표준이 정의한 DPB보다 더 많은 프레임 버퍼를 두고 디스플레이를 위한 Bumping 과정을 지연시켜, 인트라 프레임에 모자란 디코딩 성능을 이후 인터 프레임에서 보상하는 방식(Display Buffering)을 통해 동작 클럭을 감소시킨다. 하지만, 인트라 프레임에서 소모되는 시간이 너무 길어지면 이를 위한 DPB 개수가 증가하여 DRAM 사용량을 늘릴 수 있다. 이는 완제품 제조 원가에 영향을 미치는 부분이므로, 인트라 프레임을 복호화하는데 소모되는 시간을 줄이는 것은 매우 중요하다.

따라서, VVC 표준에서는 가장 많은 Bin이 발생하는 잔차(Residual) 신호에 대하여, 전체 Context Bin의 개수가 Transform 블록 크기의 1.75배를 넘지 않게 하는 제약을 정의하였다. 또한, 잔차 신호 부호화를 할 때 Context Bin 부분과 Bypass Bin 부분을 그룹화하여 연속적으로 발생하는 Bypass Bin의 개수를 증가시켰다. 이는 Bypass Bin의 경우, 여러 개의 Bin(대개의 경우 4 Bin/Clock)을 하나의 동작 클럭에 동시에 처리할 수 있다는 점을 이용하는 것으로, 이를 통해 엔트로피 복호화에서 소모되는 동작 시간을 획기적으로 줄일 수 있다. Bypass bin의 경우 Context bin과는 다르게 별도의 연산이 필요 없고, context 유도 및 업데이트 과정이 없기 때문에 하나의 동작 클럭에 여러 개의 Bin 결과를 추출하는 것이 가능하다. 현재 VTM 6.0 기준으로는 Transform-skip일 경우는 Transform 블록 크기의 2배, Transform-skip이 아닐 경우는 Transform 블록 크기의 1.75배 수준으로 Context Bin이 발생하도록 제약이 정의되어 있는데 향후 표준화 과정에서 Transform skip에 대해서도 Transform 블록 크기의 1.75배 수준의 Context Bin이 발생하도록 잔차 신호 부호화 방법이 정의될 가능성이 매우 높다.

2. 인트라 블록 크기 제약

비디오 복호화기 구현시 인트라 블록 역시 동작 클럭 선택에 영향을 많이 미치는 블록이다. 인트라 블록은 알고리즘 특성상 해당 파이프라인의 결과를 다음 파이프라인의 입력으로 사용하는 피드백 루프(feedback loop)가 존재하므로 병렬 처리가 불가능하다. 따라서, 인트라 예측 자체보다는 이를 위한 전후 과정(neighbor load, filtering, update 등)에 소모되는 시간이 전체 동작 시간에 더 큰 영향을 미친다. 또한, 이는 블록의 크기가 작을수록 더 큰 영향을 가지므로, 현재 VVC 표준에서는 인트라 블록

에서 4x4보다 작은 4x2, 2x4 그리고 2x2 블록은 발생할 수 없도록 SCIPU라는 개념을 도입하였다. Dual 트리의 경우, 각 블록의 최소 크기가 4x4로 정의되어 있기 때문에 4x4 미만의 블록이 생길 수 없지만, Common 트리의 경우 밝기 신호 기준으로 8x8 미만 블록이 생길 수 있으므로, 색차 신호에 대해서 4x4보다 작은 크기의 블록이 생길 수 있다. 따라서, Common 트리에서 QT/BT/TT 분할에 의해 발생하는 블록 크기가 8x8 미만일 때는 다음과 같은 제약이 발생한다. 1) SCIPU로 정의되는 블록의 하위 블록들은 모두 다 동일한 예측 모드(인트라/인터)를 가진다. 2) SCIPU로 정의되는 블록이 인트라 블록일 경우는, 해당 트리를 Dual 트리로 분할하여 처리하고 각각의 트리에서 4x4 미만의 블록은 생성되지 않는다.

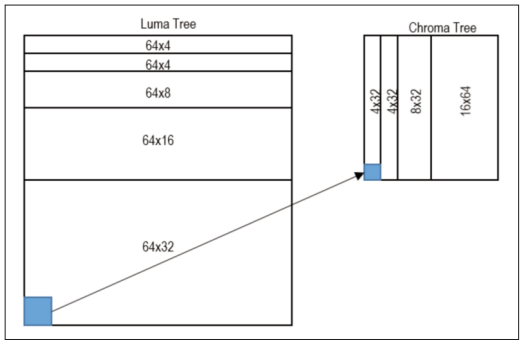


〈그림 7〉 SCIPU 조건에 의해 Dual Tree로 분할되는 예

〈그림 7〉은 이를 보여주는 그림으로, 16x8 블록에서 Ternary 트리가 적용되어 4x8 블록이 생성되고, 이들이 인트라 블록일 경우에는 밝기 신호와 색차 신호는 별도의 트리로 분할된다. 이는 기존 HEVC 대비 작은 크기의 인트라 블록 생성을 방지함으로써, 작은 인트라 블록 처리를 위한 동작 클럭 증가를 방지한다. 이와 동일한 관점에서, ISP(Intra Sub-Partition) 적용시 ISP로 분할되는 픽셀 수는 16 픽셀 이상이어야 하며, ISP 적용시 1xN 이나 2xN으로 Transform은 가능하나 예측(Prediction)은 사용할 수 없는 제약이 존재한다. 이는 일반적인 하드웨어 설계에서 인트라 블록의 결과를 행 단위로 여러 클럭에 나누어서 저장하는 특성에 기인하는 것으로, 1xN 또는 2xN 블록을 처리할 때 소모되는 클럭이, 4x4 블록 대비 많아지는 것을 대응하기 위한 제약이다.

3. Luma/Chroma Dependency 제약

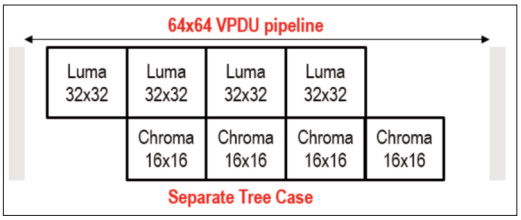
기존 비디오 코덱 표준들은 대부분 밝기 신호와 색차 신호를 병렬로 동작할 수 있도록 채널간 상관관계는 이용하지 않도록 표준을 정의하였다. 하지만 VVC에서는 성능 향상을 위하여 밝기 신호와 색차 신호간의 상관 관계를 활용하는 CCLM(Cross-Component Liner Model)과 LMCS(Luma Mapping with Chroma Scaling)라는 부호화 도구가 사용된다. 따라서, 색차 신호를 처리하기 위하여 밝기 신호의 처리를 기다리는 지연(Latency)이 발생할 수 있다. 현재 VVC 표준에서는 이런 문제를 최소화하기 위한 제약들 역시 정의하고 있다.



〈그림 8〉 Dual Tree에서 CCLM 적용시 Latency 문제

〈그림 8〉은 CCLM이 Dual 트리와 함께 동작 할 때 발생할 수 있는 지연 문제를 보여준다. 〈그림 8〉과 같이 트리가 분할되면, 첫번째 색차 신호 트리의

마지막 블록 처리를 위해서는 밝기 신호 트리의 마지막 블록이 처리가 완료되어야 한다. 이럴 경우 VPDU 기준으로 색차 신호 전체를 처리하는 시간만큼의 동작 시간의 증가가 불가피하다. 따라서, 현재 VVC 표준에서는 Dual 트리에서 CCLM이 적용될 경우, 32x32 밝기 신호만 처리가 완료되면 색차 신호의 CCLM을 시작할 수 있도록 제약이 정의되어 있다.



〈그림 9〉 DualTree에서의 CCLM 처리 예

〈그림 9〉는 해당 제약이 존재할 때 CCLM의 파이프라인 예로써, VPDU 기준으로 밝기 신호 동작 완료 후, 16x16 크기의 색차 블록을 동작하는 시간만큼의 지연이 발생하고, 이는 VPDU 전체 처리 시간의 6.25% 정도에 해당한다.

또한 LMCS에서는 색차 신호에 대한 잔차 신호를 스케일링 하는 과정의 입력으로 사용하는 밝기 신호의 평균 값을 현재 색차 신호와 같은 위치에 존재하는 밝기 신호로부터 유도하지 않고 VPDU 경계의 밝기 신호의 복원 픽셀로부터 계산한다. 이는 LMCS에서 밝기 신호와 색차 신호의 병렬 처리를 가능하게 하는 것으로 이를 통해 Luma/Chroma Dependency에 의한 지연을 최소화할 수 있다.

Ⅳ. 메모리 대역폭을 고려한 특성

서론에서 이야기한 바와 같이, 현재 대부분의 영상 기기들은 4K 영상 재생을 지원하며, 최근에는 8K TV가 출시됨에 따라, 8K 영상 재생을 위한 비디오 복호화기 구현도 필요하게 되었다. 이와 같은 영상 해상도의 증가는 SoC에서 요구하는 메모리 대역폭을 증가시켜 DRAM 사용 개수를 증가하게 만든다. 이는 완제품 제조 원가 측면에서 복호화기 면적 자체보다는 복호화기의 메모리 대역폭이 더 큰 영향을 미칠 수 있음을 의미한다. 따라서, 신규 코덱 표준 제정 단계에서 메모리 대역폭에 대한 고려가 반드시 이루어져야 하며, VVC에서도 이런 활동의 일환으로 메모리 대역폭을 HEVC 수준으로 억제하는 알고리즘 및 제약들이 채택되었다. 물론, 이런 제약들에도 불구하고 VVC 표준 특성상 작은 블록 단위의 움직임 보상이 기존 표준 대비 빈번하게 발생할 가능성이 높아서 평균 메모리 대역폭이 증가하는 것은 피할 수 없을 것으로 추정된다.

1. 인터 블록 크기 제약

비디오 복호화기 구현시, 메모리 대역폭의 대부분은 움직임 보상 과정에서 발생한다. 이전 장에서 언급된 바와 같이 움직임 보상을 위해서는 interpolation 과정을 고려해서 실제 블록 크기 대

〈표 3〉 블록 크기별 움직임 보상를 위한 입력 픽셀 수

Block 크기	제약이 없을 경우 입력 픽셀 및 비율	제약이 있을 경우 입력 픽셀 및 비율
4x4	242 (15.12배)	불가능
8x4	330 (10.31배)	165 (5.16배)
4x8	330 (10.31배)	165 (5.16배)
8x8	450 (7.03배)	450 (7.03배)
16x16	1058 (4.13배)	1058 (4.13배)
32x32	3042 (2.97배)	3042 (2.97배)
64x64	10082 (2.46배)	10082 (2.46배)

비해서 더 많은 입력 픽셀이 필요하고, 블록 크기가 작아질 수록 실제 출력 픽셀 대비 입력 픽셀의 비율이 크게 증가하게 된다. 따라서, VVC에서는 일정 크기 이하의 인터 블록의 발생을 불가능하게 하거나 또는 양방향 예측을 불가능하게 하는 제약이 존재한다. 〈표 3〉은 블록 크기 별로 이러한 제약이 존재할 때와 존재하지 않을 때 필요한 입력 픽셀 수 및 그 비율을 보여준다.

〈표 3〉에서 보듯이 만약 4x4 블록의 양방향 예측을 표준에서 허용할 경우 이를 위해 필요한 입력 수는 11x11x2로 242 픽셀이 되어 출력 픽셀 대비 약 15.12배의 입력을 필요로 한다. 따라서, 4x4 인터 블록을 허용할 경우 메모리 대역폭이 급증하는 것을 회피할 수 없다. 따라서, 현재 VVC 표준에서는 4x4 인터 블록을 불가능하게 하고, 8x4나 4x8 블록에서는 단방향 예측만 허용하여 VVC의 Worst Case 메모리 대역폭을 HEVC 수준으로 제어할 수 있도록 하고 있다.

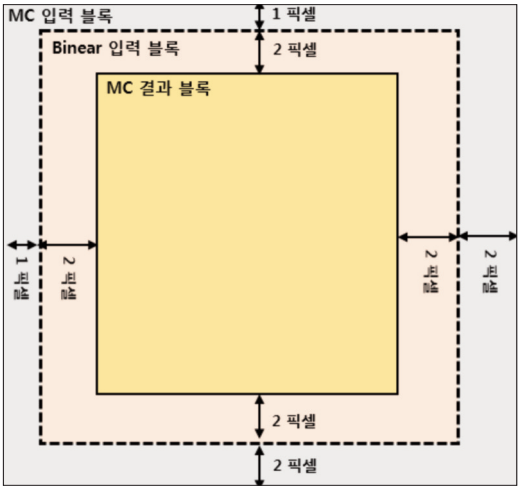
2. DMVR, Affine 제약

VVC에서는 인터 블록의 예측 성능 향상을 위하여 DMVR과 Affine 움직임 보상이 사용된다. VVC에서는 이런 새로운 부호화 도구들이 기존 움직임 보상 대비 더 많은 대역폭을 요구하지 않도록 부호화 방법 및 제약이 정의되어 있다.

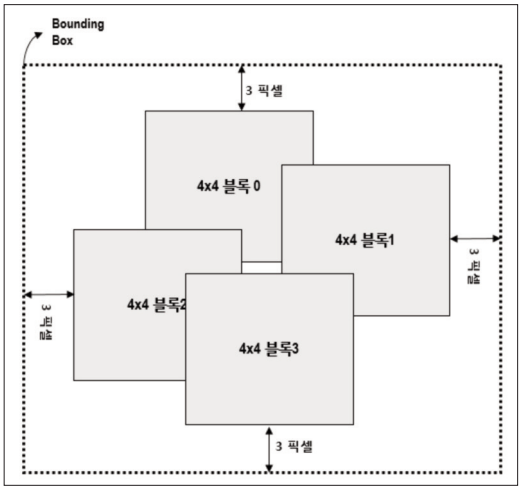
DMVR의 경우, DMVR에 의해 수정되는 MV에 의해 필요로 하는 입력 픽셀 수가 기존 움직임 보상 과정 대비 증가하지 않도록 부호화 방법이 설계되어 있다.

DMVR에 의해 수정된 MV는 원래 MV로부터 상, 하, 좌, 우 최대 2 픽셀까지 변할 수 있으므로, 움직임 보상 단계에서 방향별로 2 픽셀의 추가적인 입력을 필요로 할 수 있다. 하지만, VVC의 DMVR에서는 다음과 같은 제약을 통해 DMVR 적용시 입력 픽셀 수를 기존 움직임 보상과 동일하게 유지한다.

1) 수정된 MV가 기존 움직임 보상 대비 더 많은 입력을 필요로 할 때는 해당 입력은 패딩(Padding)으



〈그림 10〉 MC와 DMVR의 경우 입출력 픽셀 범위



〈그림 11〉 Affine MC에서 Bounding Box 개념

로 처리한다. 2) 수정된 MV 탐색을 위한 SAD 계산에서는 2-Tap Bilinear 예측을 사용한다. 이런 제약들이 사용됨으로써, VVC의 DMVR은 〈그림 10〉과 같이 기존 움직임 보상을 위한 입력 픽셀과 동일한 입력 픽셀만으로 구현이 가능하다.

Affine 움직임 보상의 경우, 입력 블록을 4x4 단위로 분할하여 서로 다른 MV로 움직임 보상을 수행하기 때문에 작은 크기의 인터 블록 발생으로 인하여 메모리 대역폭이 증가할 수 있다. VVC에서는 Bounding Box라는 개념을 도입하여 이런 문제를 해결하였다.

양방향 Affine 움직임 보상의 경우, Bounding Box는 〈그림 11〉과 같이 8x8에 해당하는 4개의 4x4 블록의 움직임 보상 영역을 사각형으로 묶은 영역으로 정의된다. VVC 표준에서는 이 Bounding Box의 픽셀 수가 기존 움직임 보상을 위한 입력 픽셀 수 이하일 때만 Affine 움직임 보상을 수행하고, 그렇지 않을 경우에는 모든 4x4 블록에 대해서, 현재 블록의 가운데 4x4 블록의 Affine 움직임을 모든 4x4 블록에 적용하도록 강제한다. 즉, Bounding Box의 크기가 기존 8x8 움직임 보상의 입력 크기인 225 이하일 때만 Affine 움직임 보상을 적용하게 한다. 단방향 예측일 경우에는 Bounding Box가 8x4와 4x8 모두에 대하여 각각 정의되고 그 크기들이 모두 165 픽셀을 넘지 않을 때에만 Affine 움직임 보상이 적용된다. Affine 움직임 보상에서는 기존 움직임 보상 대비 작은 tap 수의 interpolation을 수행하므로, Bounding Box에 의해 Affine 움직임 보상이 과도하게 적용되지 않는 현상은 회피할 수 있다.

V. 결 론

본 논문에서는 현재 VVC 표준(VVC CD, VTM6.0)에 포함된 하드웨어 복호화기 설계를 고려한 부호화 도구 및 제약 들에 대하여 살펴보았다. VVC의 경우 기존 HEVC 대비 증가한 부호화 도구의 숫자 및 복잡도, CTU 블록 크기, 그리고 SCC, 360도 비디오, Sub-Picture, HDR, RPR(Reference Picture Resampling)과 같은 다양한 기능 요구 사항들을 고려했을 때 하드웨어 복호화기의 면적 및 구현 난이도 증가가 예상된다. 하지만 표준 개발 단계에서 하드웨어 복호화기 구현을 고려하여 여러 가지 제약이 정의되고 알고리즘 최적화가 진행되었기 때문에, 실제 하드웨어로 구현하여 제품 적용이 가능한 수준의 표준이 완성될 것으로 예상된다. 따라서, MPEG VVC는 향후에도 기존 MPEG 코덱들처럼 시장에서 널리 사용될 수 있을 것으로 판단된다.

참 고 문 헌

[1] JVET-O2001, “Versatile Video Coding (Draft 6)”
[2] VTM 6.0 Reference Software, “https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM”

필 자 소 개

이 상 헌

– 2001년 3월 ~ 2005년 2월 : 서울대학교 전기공학부 학사
– 2005년 3월 ~ 2011년 8월 : 서울대학교 전기공학부 박사 (석박통합)
– 2011년 9월 ~ 현재 : LG전자 CTO SIC센터